Informix Dynamic Server

Fun and Games: The Data Server for Interactive Entertainment

Author: Rajesh Nair
rajeshn@us.ibm.com

INTRODUCTION

Massively Multi-Player Online Games (MMOGs), virtual worlds, and social networks are in the process of revolutionizing interactive entertainment. It is no coincidence to be talking about all three of these segments together. There is a convergence happening among the three that opens up many new and interesting opportunities from technology, marketing, and gaming experience perspectives.

MMOGs and virtual worlds are characterized by persistent interactive environments that continue to exist irrespective of whether users are currently active in them. With advances in technology, game and virtual world companies have the ability to deliver rich, deep game play with an inspiring immersion experience. Popular MMO games and virtual worlds can have several hundred thousand concurrent users. Throughput performance and latency become essential metrics for companies in this industry.

The MMOG environment is demanding increasingly complex information management. Large amounts of data are being stored and utilized during game-play. Not only does this data persist and change as a result of game play, but data about the game and its players is also being gathered and used to help to improve the game and grow market share. For these reasons, data servers are now considered table stakes for CTOs in the games industry. Icarus Studios for instance, considers game data management and their use of a database engine a key differentiator in providing a better experience. The database engine that Icarus Studios chose to adopt is Informix Dynamic Server (IDS), which lies at the heart of the innovative Icarus technology suite.

IDS is based on a unique and proprietary architecture that makes it particularly well suited for the MMOG industry. From a performance and latency standpoint, one of the greatest strengths IDS brings to the industry is its inherently scalable and multi-threaded architecture. This architecture is built on technology called the Dynamic Scalable Architecture (DSA), and it enables IDS to fully utilize hardware resources and scale vertically. IDS includes built-in multi-threading and parallel processing capabilities -- which allow it to process similar types of database activities in parallelized groups instead of as discrete operations. IDS 11 also offers powerful yet flexible options for high availability and scale. Further, with minimal administration requirements and built-in autonomics, the IDS total cost of ownership (TCO) is exceptionally low among database offerings.

With IBM hardware and services already being the most popular choice for extreme performance requirements demanded by this industry, IBM's high performing Informix Dynamic Server presents the best choice and value for game data management for existing and new markets in this industry.

THE IDS VALUE PROPOSITION

Simply put, IDS is the leading data server choice for the interactive entertainment industry because IDS:

- is extremely fast,
- has a small footprint,
- is very easy to manage,
- scales seamlessly,
- is cost effective, and
- is highly resilient

IDS is designed from the ground up to be integrated and optimized with low latency systems that demand scale yet require transparent and low-cost administration and management overhead. IDS 11 capabilities also help achieve "five 9's" (99.999%) of high availability allowing for maximum uptime for critical databases.

There are two key segments in the interactive entertainment lifecycle: Game Production and Game Execution. IDS is ideally suited for and provides value in both of these areas.

IDS IN GAME PRODUCTION

Game production cycles are very similar to software development cycles in other industries – in which a product moves from concept through design, development, QA, release, and maintenance. Key differentiators in game production cycles are:

- The vast number of assets that are generated
- The high degree of collaboration required

Hundreds of thousands of assets or game objects are pulled together to seamlessly represent the game world. Using a data server to manage this data plays a critical role in rapid application development, reducing costs and improving time to market. Key IDS capabilities that bring value to game production include its broad support of application development interfaces, ability to scale-out on low cost hardware, high performance, flexibility in defining new user defined types and routines, ease of management, and therefore exceedingly low total cost of ownership.

Even larger than the challenge of game production is the ability to successfully manage game operation and execution. To survive and thrive in the market, online game companies must define, meet, and monitor metrics related to performance, scale, and availability. Whether a game provider can meet or exceed these metrics can make the difference between success and failure.

IDS IN GAME EXECUTION

The online game execution or operational environment ranks among the most complex of any runtime infrastructure. All games consist of a client that communicates to the infrastructure running the virtual world. Between them, they manage huge amounts of data of various types, hundreds of thousands of users, and render complicated graphics and animation.

A typical server side deployment consists of several different types of servers arranged in a tier to manage various aspects of the game:

- login servers that manage authentication, verification, billing
- game servers that host the core game engine and related applications
- communications servers that manage user interaction in the form of text and voice chats
- simulation servers that create and model workloads
- data servers that manage game and user data

Millions of geographically-distributed people interacting with a virtual world require a tremendous amount of communication and content management to keep them occupied and engaged. A common model for addressing extreme volume and concurrency is based on “shards”. Shard architecture consists of duplicated game processes across servers or server clusters so that every shard runs its own copy of the game world and handles fixed user traffic. Each shard also typically has its own instance of the game database. It’s like duplicating planet Earth many times. You can communicate with everyone who lives on “your Earth” but not on the other Earths. If you want to communicate with someone on another Earth, you have to give up everything on your Earth and move to the other one!

Shards are not seamless experiences for players or for the company operating the game, since at any given point in time there can be state differences across the different shards. If there are hundreds of shards, it also becomes a maintenance challenge. Upgrades to game software require synchronization across all of the shards. Hardware upgrades also need to be synchronized.

Figure 1 below shows a very simplified visual of shard architecture. Whether to use shard architecture is decided when the game is designed. A shard is typically implemented as a cluster of game engines or is sometimes a single high performance machine hosting a single game engine.

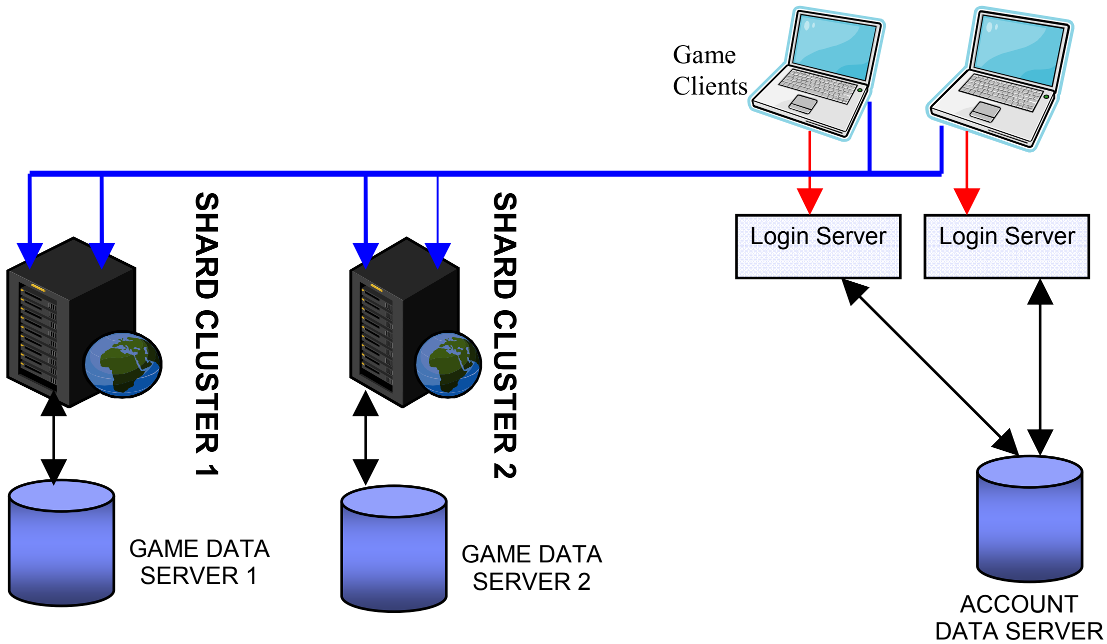


Figure 1 – Simplified Shard Model

PERFORMANCE DEMANDS ON THE DATA SERVER

There are a variety of performance elements that must be considered when choosing a data server. The ability of the infrastructure to keep game play responsive and at the same time support a growing user base is paramount to a game's success. When a company puts together a plan to analyze different components for its infrastructure, common factors include numbers of users handled, milliseconds of response time, and infrastructure capability (such as ability to handle shardless one-world scenarios, for example).

In order to determine which data server capabilities are needed, it is important to know the key performance parameters for the game and what a company's expectations are for those parameters. Significant parameters include:

- Target number of total players
- Target number of total players per geography (based on players who will buy and play the game in different regions of the world)
- Target number of concurrent players
- Target number of concurrent players per geography
- Target number of concurrent players per shard (for a shard-based model, this will be critical in determining hosting hardware)
- Average amount of data transmitted per second per player – this is critical in understanding network infrastructure requirements and data server requirements
- Target size of world data – refers to the expected amount of data that will be stored in databases

- Target data size per player – amount of persistent data a player will generate during game play
- Target latency/response time – the expected response time for a roundtrip involved in game play updates, typically expressed in milliseconds.

It is rare that a company will be able to provide reasonable estimates for each and every metric listed above. However, obtaining even a few key estimates helps a great deal.

A significant number of the performance requirements are influenced by machine hardware and network components. Also, while many companies rely on the shard model where the game world is divided into duplicates, other companies are looking at providing a "shardless" one-world experience. Examples of shardless experiences include EvE Online, developed and published by CCP, and Second Life from Linden Lab.

Shard vs. shardless architecture influences data server design and configuration. With shards, each game shard typically has a copy of the database instance which in turn means that each database instance will handle limited number of concurrent sessions. A shardless deployment presents a different set of options for database instance configuration.

Regardless of the architecture, a critical metric is ***latency***. Latency can be defined as the time it takes to complete one transaction. A transaction is a round trip from the game client to the game server and back that results in some refresh of the game world state. ***The amount of time it takes the data server instance to service requests from the game server is part of this response time - and only part of it***. Consider some of the other pieces that influence latency:

- The time it takes for the game server to process incoming requests
- The way the game services package requests to the data server instance – non optimal packaging of transactions can increase latency
- The time it takes for data to be read from the media

In the game world, latency requirements are one of the most important performance factors. Latency is impacted by design decisions that lay at the heart of the game engine and client communication systems. Latency is also influenced by the distribution of elements in the network and of course, the speed of the Internet.

Transactions in online games are large in volume but small in size – simple updates and queries that happen very frequently. Traditionally, database writes require more time to execute than database reads. However, with effective and advanced use of memory and parallel systems built into IDS –read and write latencies can be very close. As mentioned earlier, one of the greatest strengths that IDS brings to the industry from a performance and latency standpoint is its inherently scalable and multi-threaded architecture built on proprietary Dynamic Scalable Architecture (DSA) technology.

The IDS architecture is based on a set of processes called virtual processors (VPs). There are multiple types of virtual processors to handle specific tasks. Each VP is multi-threaded and can handle multiple tasks and adjust dynamically to the demand of the system by adding or removing threads as it sees fit. This threading model has been designed specifically to optimize database operations.

The IDS threading architecture also manages user connections and the parallelism of SQL queries or other database operations. This use of threads provides the optimal foundation for performance and scalability. Threads offer much higher performance than similar operations on processes.

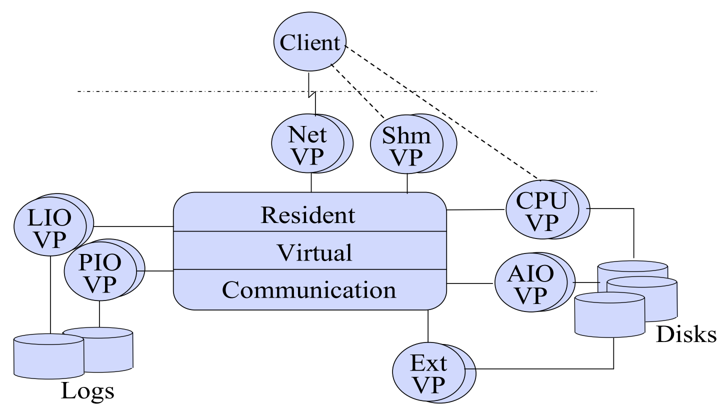


Figure 2. – IDS Virtual Process (VP)-based DS Architecture

Taking IDS a Step Further

There's another level of scale and performance that CTOs consider as they envision the exponential future of their games market. How do you sustain performance (throughput and latency) as a game gets popular and grows its concurrent user base over time? Even with minimized latency and DSA disk I/O optimization, there's an even faster scenario that results from leveraging two exciting technologies alongside IDS:

- In-memory databases
- Solid state disks

In-memory databases reside and process transactions entirely in memory. Without the need to perform disk I/O and cache data, in-memory databases can process data in real time.

Eventually, data will need to be persisted to disk and used for business purposes. IDS with its blazing fast OLTP and scale combined with IBM's in-memory solidDB product provide an integrated extreme transaction processing model for gaming companies with very low latency requirements that need to be sustainable over a long term.

Solid state disks (SSDs) offer additional advantages. The most popular technology behind solid sate disks is nonvolatile flash memory. The absence of moving parts eliminates seek time and other electro-mechanical delays associated with traditional disk drives. CCP's EVE online is a prime example of a game system that claimed a 4000% performance improvement when moving from traditional disks to SSD storage. SSDs do not present any new requirements for data servers.

A combination of either an in-memory database like solidDB with IDS or SSD with IDS can yield extraordinary performance gains.

MANAGING SCALE AND AVAILABILITY

Handling growth is always a challenge. While it's good for the business, it's hard on the infrastructure! The ability to manage scale has a huge impact on the business. The consequences of not being able to scale over time include drops in performance or potentially a complete lockout of users – significant issues for paying subscribers. The same holds true if game play were to be disrupted due to server failure or other events.

In a shard architecture, scale is managed by replicating host systems – this means adding new identical hardware that runs its own game server and database instance. As existing game servers start to reach peak loads, a new node is quickly added.

The biggest business challenge with shards is estimation. Companies have to estimate at what point to cap each shard at and then estimate the number of shards needed. If they underestimate, they risk performance and user lockouts. If they overestimate, they have unused hardware and sunk costs.

Further, shards require the cost of supporting infrastructure. As shards grow in number, a "patch" server needs to be setup to perform new patch upgrades. Second, additional support staff maybe needed to monitor, analyze and maintain the different nodes. Last but most significant, shards inhibit true seamless game play. Users on a shard cannot play with users on another shard.

Some companies today are therefore attempting a shard-less architecture. In a shard-less architecture, there is a single instance of the game world and every user can truly interact and play with any other.

While IDS can scale up and take advantage of additional hardware processing power and memory on a single node, the built-in high-availability suite in IDS 11 provides optimal performance for a shard or shard-less world.

IDS 11 high availability suite consists of the following configurations:

- Continuous Availability Feature (CAF) using Shared Disk Secondary (SDS) technology: SDS configurations allow IDS instances to quickly and easily scale in a cluster environment with multiple instances accessing data from a shared disk. While one instance is designated as the primary instance, any of the SDS nodes can be quickly promoted to take over as primary if the primary becomes unavailable.
- Remote Standalone Secondary (RSS): RSS configurations consist of multiple geographically distributed IDS instances that maintain their own copies of data providing high availability and disaster recovery. Any RSS node can also be utilized for additional workload balancing.
- High Availability Data Replication (HDR): HDR has been the cornerstone of IDS availability for several years. An HDR configuration consists of a primary and a secondary node. In the event of the primary being unavailable, the HDR secondary can provide uninterrupted access to data. The HDR primary and secondary can be configured to be in SYNC mode making HDR the best choice for high transactional integrity requirements during a failover.
- Enterprise Replication (ER): Configuring IDS for ER provides a host of flexible and powerful options to distribute or consolidate data. With IDS ER, data can be distributed across instances in either a primary-target (master-slave) or update anywhere model (master-master).

All secondary servers in these configurations are both readable and writeable thus enabling quick, efficient and reliable scalability as users and data grow.

For efficient and automatic management of secondary instance clusters and for automated failover, IDS 11 introduces the Connection Manager (CM). The Connection Manager is a lightweight process that uses rules or Service Level Agreements (SLAs) to manage a cluster of secondary and failover instances. The IDS CM dynamically routes client application connection requests to the most appropriate server in a high-availability cluster. The Connection Manager connects to each of the servers in the cluster and gathers statistics about the type of server, unused workload capacity, and the current state of the server.

Now let’s look at how IDS capabilities can help meet the requirements of scale and availability for 3D virtual environments.

Here is a generic high level database design for online games:

- Account database – contains data on subscribers/players. This database is typically maintained on a separate instance. Login servers use this database to perform authentication as well as to insert new subscriber information. This account database has a very high volume of read activity and relatively few writes (new user registrations or profile updates)
- Game world database – holds world specific data such as layout, Non-Playing Characters (NPCs), game strings, etc. This is a read only database
- Player database: stores changes to the player’s character, skill, level, and any milestones reached. This database is updated very frequently

- Logging Database –used for administrative logging. This is primarily used to track player movements, behavior (for example, are they cheating?) and potential troubleshooting data. Because of the high volume of writes, this is often maintained as a separate database.
- Reporting and stats database – contains relevant data from the logging database that is used for business analysis, troubleshooting, marketing etc. This is largely a read-oriented database.

Some game developers load all of the world's read-only information into memory to optimize for the large number of read requests.

Note that this as a very generic scenario for illustration purposes. Individual providers and developers would design aspects of this differently.

In order to have an optimal IDS configuration for game sessions it is first necessary to understand:

- the database design
- usage patterns for each database – for example, the account database and world database would be read-oriented databases with very frequent access
- the size of the world and player data
- the game server architecture – clustering, geographical distribution

Based on our generic scenario,, following are some examples of managing scale and balancing workload using IDS capabilities. Please remember that these are examples and actual configurations will be driven by specific customer requirements.

FAILOVER AND WORKLOAD DISTRIBUTION WITH HDR AND RSS

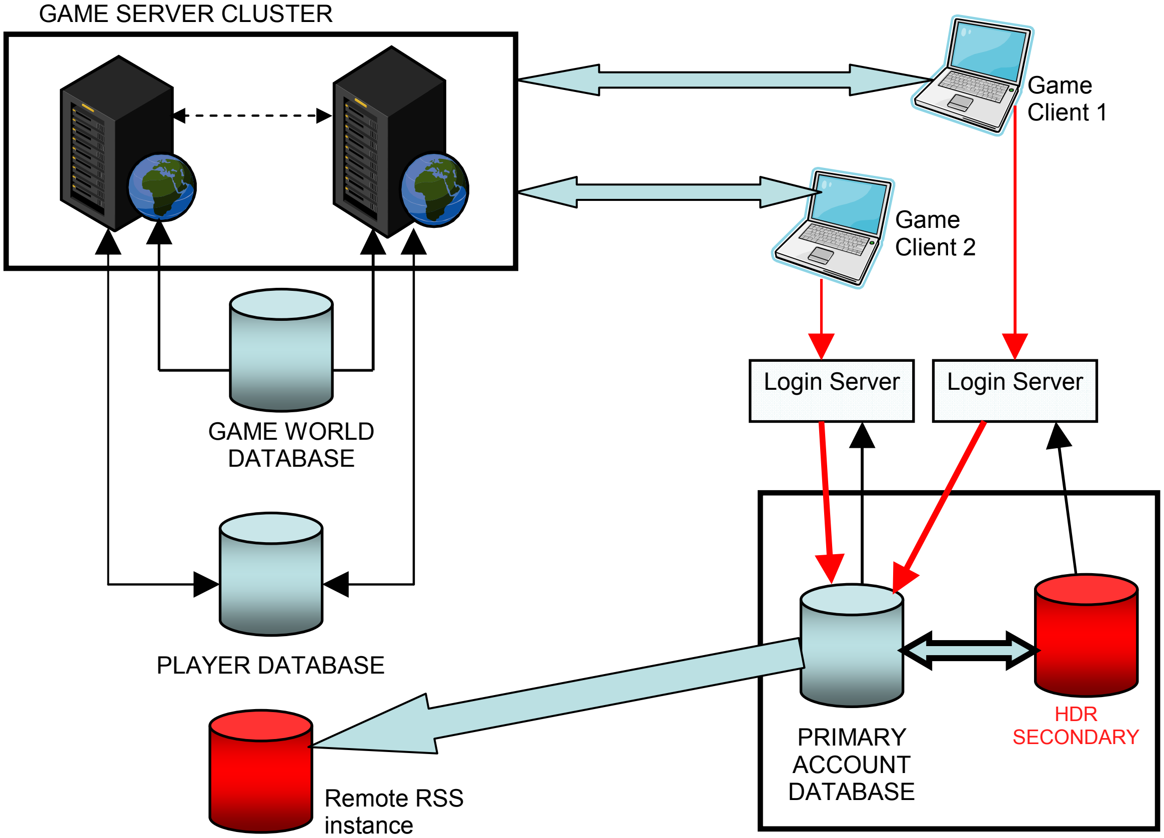


Figure 2

As noted earlier, the account database instance services login servers. The login process is a critical first step in the game experience. If not executed properly, customers can experience long wait times and complete lockouts on login attempts during peak cycles or server outages on highly popular games. Not only does this annoy paid subscribers, but it can also cause bad PR for the games.

The login process is a very read-oriented intense activity with occasional new user registrations. Configuring an IDS HDR secondary for the account database not only provides a reliable failover option, but also provides additional load-balancing by allowing some traffic to be routed to the secondary. An additional IDS RSS node provides additional business continuity at a geographically remote location. An RSS node can also be configured to handle additional workload requirements if necessary.

FAILOVER AND WORKLOAD DISTRIBUTION WITH SDS

The Player Database is one of the most active databases during game play. It is highly transactional in nature. The player database can be very large.

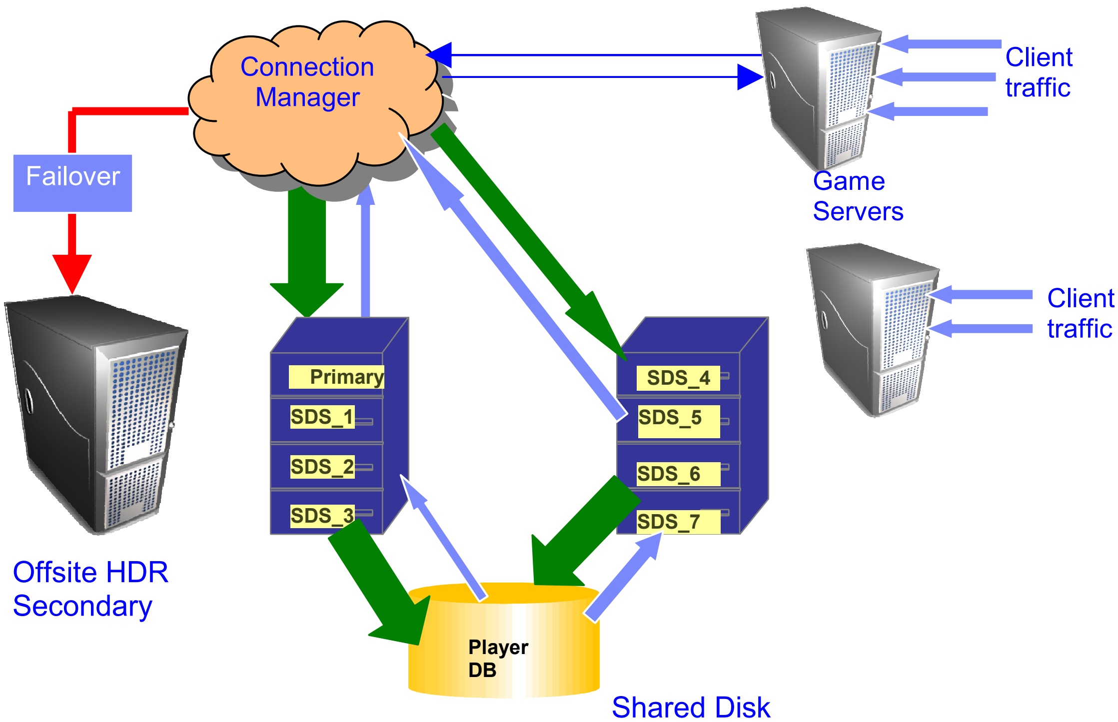


Figure 3

In the configuration above, multiple game servers are configured using IDS SDS instances on a blade server with data managed on a shared disk. Since SDS nodes can be added as needed, the system can grow as demand requires. If an SDS node fails the system continues to run, although fewer nodes will be available for processing requests. If the primary node fails then one of the SDS nodes is automatically promoted as the new primary and the remaining SDS nodes automatically reconnect to the new primary.

WORKLOAD OPTIMIZATION WITH ENTERPRISE REPLICATION

One activity where ER can help optimize performance and workload is “in-game logging”. In-game logging is a critical support function that helps game companies capture reporting information including:

- troubleshooting information
- behavioral information to keep game play fair
- administrative information such as login/logout times
- other miscellaneous data to be mined for marketing or game improvements

The degree, depth and persistence of data captured vary from company to company depending on specific business needs.

Hundreds of thousands of concurrent players generate a lot of logging activity. Rather than using a centralized database instance to handle all this activity, it's preferable to have each game server cluster designed to perform updates locally. With IDS ER, this design can easily be achieved by replicating out the required tables to IDS instances set up locally with game server clusters as demonstrated in figure 4 below.

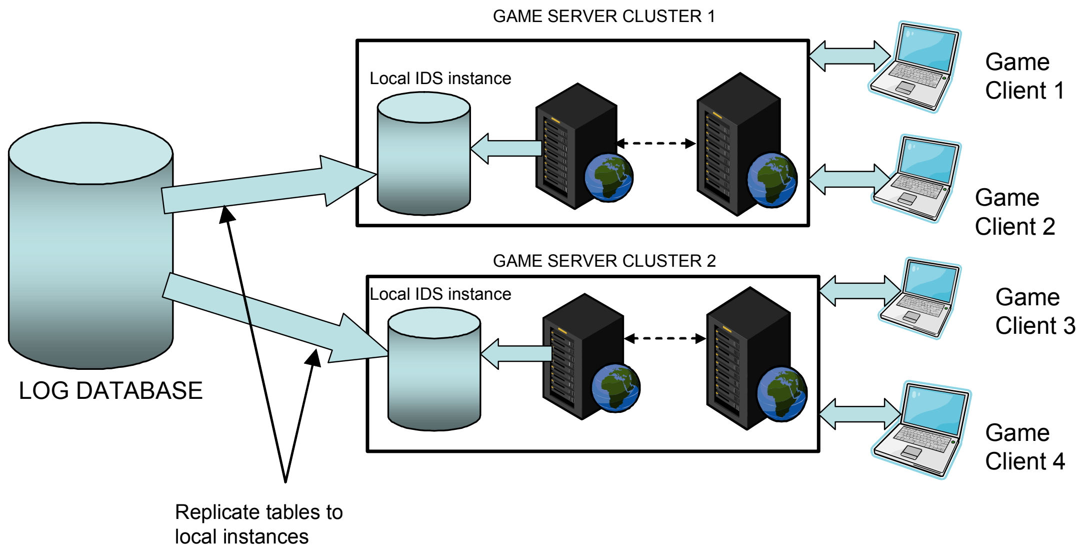


Figure 4

Data from this database can also be offloaded to a separate database for reporting and data mining.

Another very useful application of ER is to optimize lookup activity by replicating frequent lookup tables to local instances. For example, during the login process several servers are typically dedicated to manage authentication and verification. As alluded to earlier, the login process is a critical support function and is the first activity that a player performs before actually playing the game. Hundreds of thousands of players attempting to login can generate a lot of lookup activity and can impact system performance.

With IDS ER, once again, the required lookup tables can be replicated to local IDS instances on the login servers to optimize performance.

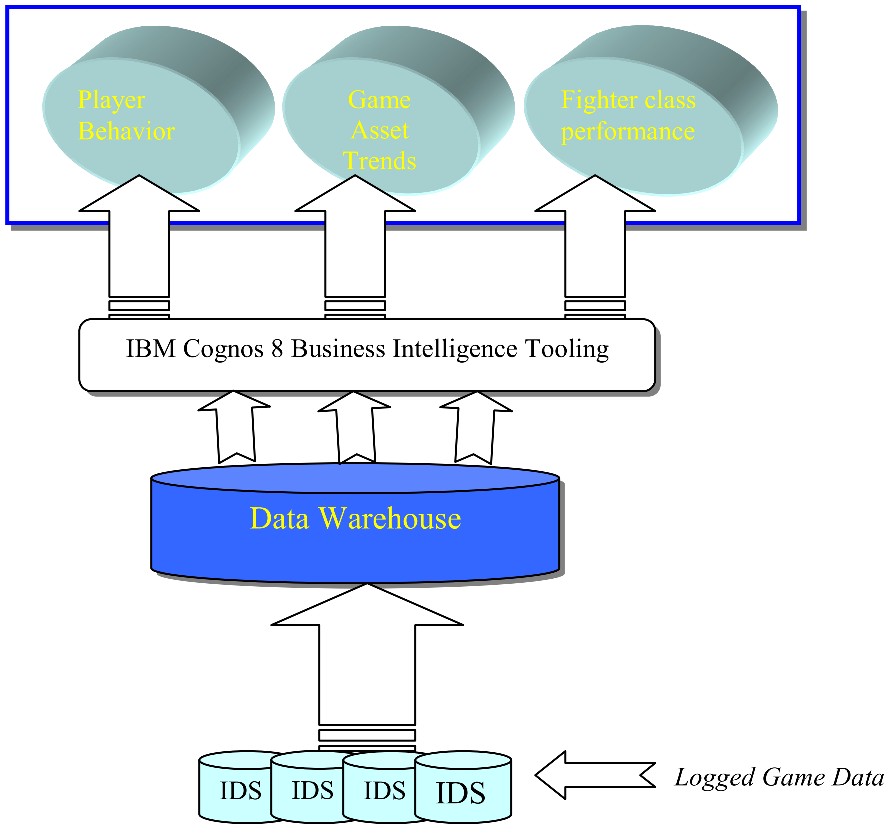
DATA MINING

There are several reasons why companies want to invest time and effort in mining the data collected during game play. Data mining can help improve game design, reduce costs, reduce or prevent fraud and consequently maximize game ROI. While official surveys and feedback mechanisms can be put in place, mining game data provides a much richer and consistent source of information for analysis, improved decision making and future game improvements.

Data mining is an important and critical business process. Like any other data management process, efficiency, performance and cost combined with the right tools for analysis are important to successful data mining. IDS in combination with IBM Cognos represent industry leading capabilities in analytics and reporting that provide maximum value for game data mining.

The quality of the data is very important for analysis and decision-making. For this reason, the entire process that leads to data mining needs to be well-planned. A lot of raw data is collected during game play. Trying to mine the deluge of raw data can be counter-productive and debilitating. Therefore, the process should include some critical steps to clean, archive and extract the appropriate data.

Eliminating data that is not relevant makes the data set cleaner and more efficient to start analysis. This data set however, still represents a lot of data, all of which may not be required for mining. Business policies for data archival should be followed and acted upon to appropriately archive this data set into a warehouse. For data mining purposes, the relevant data fields should now be extracted and transformed to a more focused data set. With the help of business intelligence tools similar to what IBM Cognos provides, invaluable insights into associations and relationships can be derived from this set.



SERIOUS GAMES

Does the application of game technology stop at games? Not according to a growing market called "serious games". Serious games are not really games, but application of game technologies to requirements and opportunities outside games.

Games are capable of being highly immersive by integrating 3D graphics, social networking and collaboration tools. This same combination of technology holds potential for industries and organizations including next generation education, advertising and corporate training solutions. Consider for example, the "Virtual Rome" project which involved the reconstruction of 320 AD Rome online. This project shows the potential of revolutionizing education by allowing students to learn Rome by "experiencing" it.

Also consider how corporations can benefit from this technology by constructing rich training solutions using scenario-based virtual worlds for example, in customer service or sales training. The same qualities of IDS that make it the leading choice for MMOs and Virtual Worlds also make it an ideal choice for these cross-industry solutions.

SUMMARY

Virtual worlds, MMO games and social networking are redefining the nature of interactive entertainment as they strive to provide customers with deep immersive experiences.

In the lifecycle of these next generation virtual realities, a lot of assets and data need to be managed. IBM's Informix Dynamic Server with its industry leading performance, scalability, high availability, low TCO and rapid application development capabilities offers the maximum value to game developers, publishers and service providers.

As virtual worlds and MMO games become more popular, CTOs are increasingly realizing the need for efficient, reliable data management as well as a highly scalable and available information architecture. While IDS can scale up and maximize the advantage of additional hardware processing power and memory on a single node, it is in combination with the scale out and availability capabilities built into IDS that virtual world and MMO companies gain a competitive advantage as their businesses grow.

In combination with IBM's solidDB in-memory database, IDS can help meet a new level of extreme performance requirements. IDS capabilities allow game designers to create a shardless architecture that is viable and efficient. IDS in combination with IBM Cognos provides industry-leading capabilities for data mining, a critical process that can give companies a competitive advantage.

Finally the technologies behind virtual worlds and MMOGs create exciting opportunities for the serious games market as well, and these companies benefit from the same IDS capabilities.

IDS is a standout product, with a proprietary virtual-procesor-based threaded architecture that makes it uniquely suited to the particular demands of the virtual games environment.